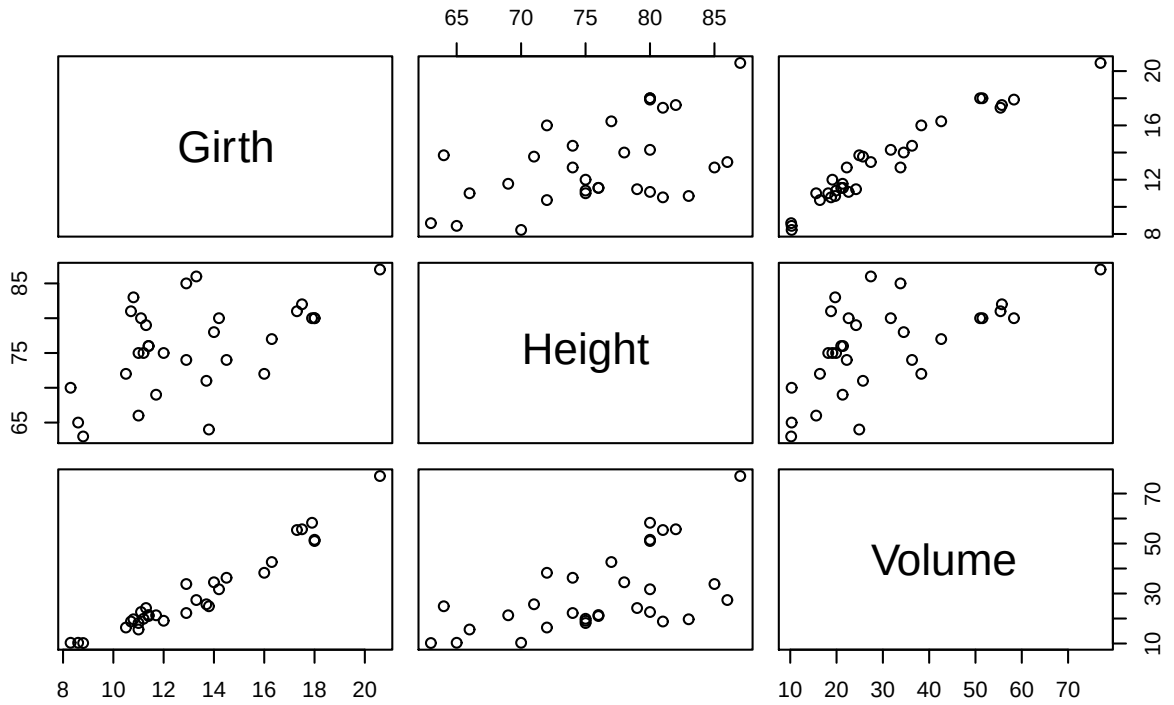
GAMs, GAMMs and other penalized GLMs using `mgcv` in R

Simon Wood
Mathematical Sciences, University of Bath, U.K.

Simple example

- Consider a very simple dataset relating the timber volume of cherry trees to their height and trunk girth.



`trees` initial `gam` fit

- A possible model is

  $\log(\mu_i) = f_1(\texttt{Height}_i) + f_2(\texttt{Girth}_i), \quad \texttt{Volume}_i \sim \text{Gamma}(\mu_i, \phi)$

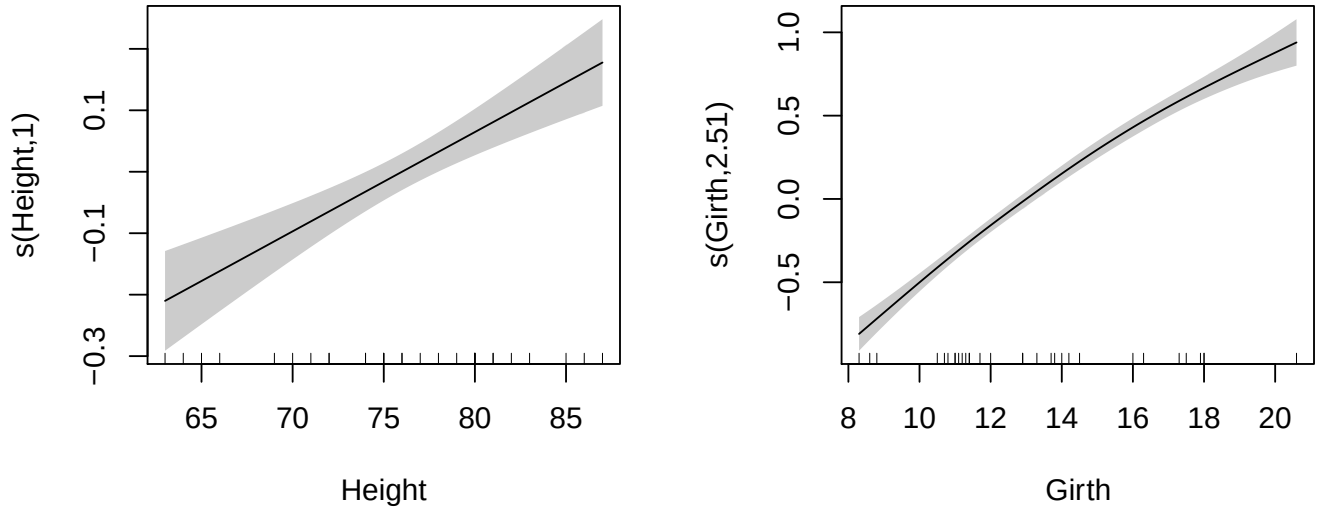
- Which can be estimated using...

```
library(mgcv)
ct1 <- gam(Volume ~ s(Height) + s(Girth),
           family=Gamma(link=log),data=trees)
```

- `gam` produces a representation for the smooth functions, and estimates them along with their degree of smoothness.

Cherry tree results

- `par(mfrow=c(1,2))`
  `plot(ct1,various options) ## calls plot.gam`



- This is a simple example of a Generalized Additive Model.
- Given the machinery for fitting GAMs a wide variety of other models can also be estimated.

The general model

- Let y be a response variable, f_j a smooth function of predictors x_j and L_{ij} a linear functional. A general model...

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\alpha} + \sum_j L_{ij} f_j + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\theta), \quad y_i \sim \mathrm{EF}(\mu_i, \phi)$$

  $\mathbf{A}$ and $\mathbf{Z}$ are model matrices, g is a known link function.
- Popular examples are ...
  - $g(\mu_i) = \mathbf{A}_i\boldsymbol{\alpha} + \sum_j f_j(x_{ji})$ (Generalized additive model)
  - $g(\mu_i) = \mathbf{A}_i\boldsymbol{\alpha} + \sum_j f_j(x_{ji}) + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\theta)$ (GAMM)
  - $g(\mu_i) = \int h_i(x) f(x) dx$ (Signal regression model)
  - $g(\mu_i) = \sum_j f_j(t_{ji}) x_{ji}$ (Varying coefficient model)
  - The list goes on...

Representing the f_j

- A convenient model representation arises by approximating each function with a basis expansion

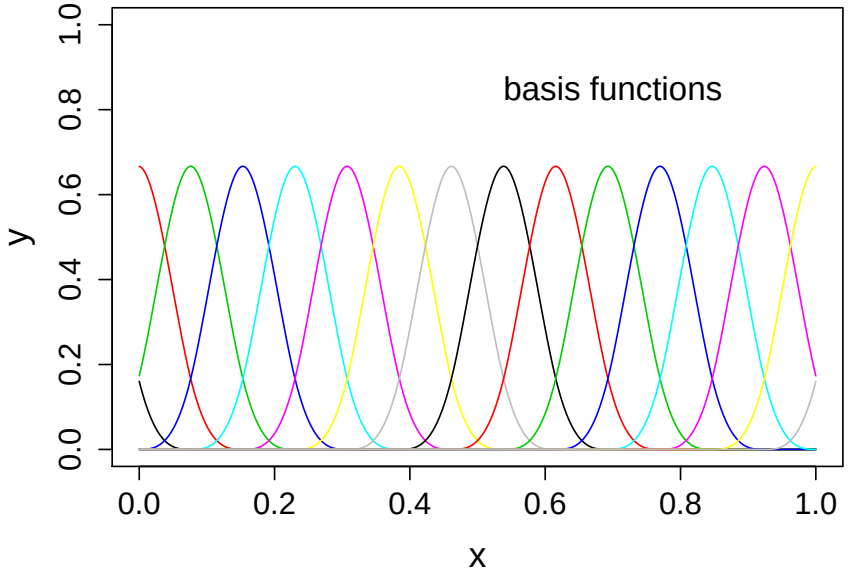
$$f(x) = \sum_{k=1}^{K} \gamma_k b_k(x) \quad (b_k \text{ known, e.g. spline})$$

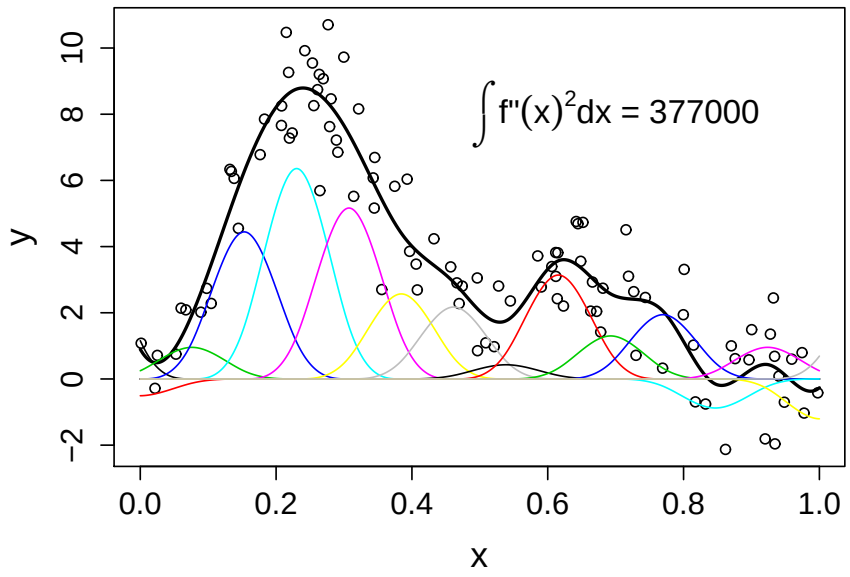
- K is set to something 'generous' to avoid bias.
- Some way of measuring the 'wiggliness' of each f is also useful, for example

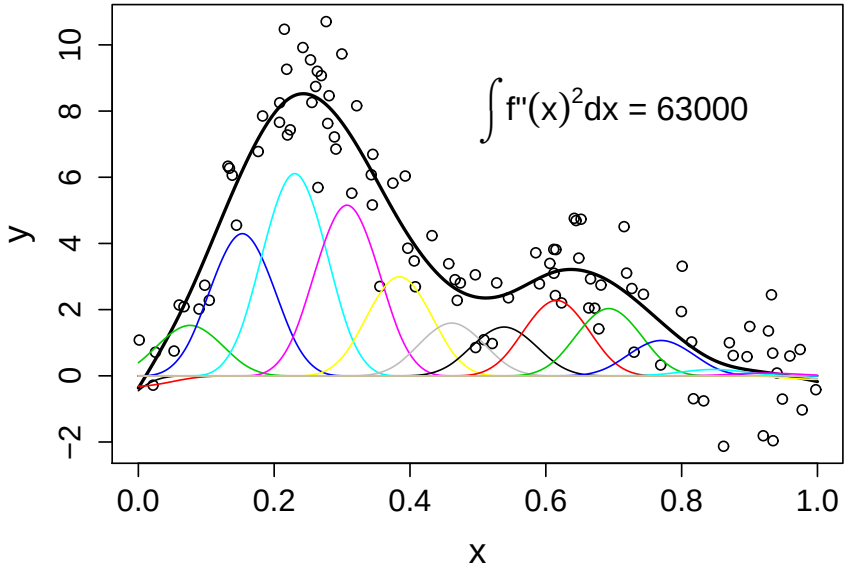
$$\int f''(x)^2 dx = \gamma^{\mathrm{T}} \mathbf{S} \gamma. \quad (\mathbf{S} \text{ known})$$

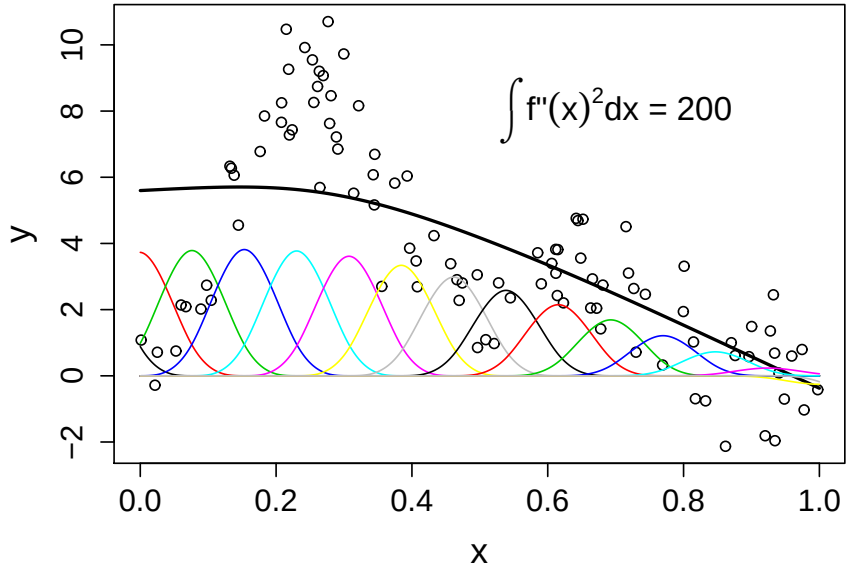
- For many models we will need an identifiability constraint, $\sum_i f(x_i) = 0$. Automatic re-parameterization handles this.

Basis & penalty example









Representing the model

- The general model is

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\alpha} + \sum_j L_{ij} f_j + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\theta), \quad y_i \sim \text{EF}(\mu_i, \phi)$$

- Given a basis for each f_j, we can write $\sum_j L_{ij} f_j = \mathbf{B}_i\boldsymbol{\gamma}$, where $\mathbf{B}$ is determined by the basis functions and L_{ij}, while $\boldsymbol{\gamma}$ is a vector containing all the basis coefficients.
- Then the model becomes ...

$$g(\mu_i) = \mathbf{A}_i\boldsymbol{\alpha} + \mathbf{B}_i\boldsymbol{\gamma} + \mathbf{Z}_i\mathbf{b}, \quad \mathbf{b} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\theta), \quad y_i \sim \text{EF}(\mu_i, \phi)$$

- Problem: if the bases were rich enough to give negligible bias, then MLE will overfit.

Estimation strategies

1. Control prediction error
   - Penalize the model likelihood, using the f_j wiggliness measures, $\sum_j \lambda_j \boldsymbol{\gamma}^{\mathrm{T}} \mathbf{S}_j \boldsymbol{\gamma}$, to control model prediction error.
   - Given $\boldsymbol{\lambda}, \boldsymbol{\theta}, \phi$, this yields the maximum penalized likelihood estimates (MPLE), $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\gamma}}, \hat{\mathbf{b}}$.
   - GCV or similar gives $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\theta}}, \hat{\phi}$.
2. Empirical Bayes
   - Put exponential priors on function wiggliness so that $\boldsymbol{\gamma} \sim N\left(\mathbf{0}, (\sum_j \lambda_j \mathbf{S}_j)^{-} / \phi\right)$.
   - Given $\boldsymbol{\lambda}, \boldsymbol{\theta}, \phi$ the MAP estimates $\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\gamma}}, \hat{\mathbf{b}}$, are found by MPLE as under 1.
   - Maximize the marginal likelihood (REML) to obtain $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\theta}}, \hat{\phi}$.

Maximum Penalized Likelihood Estimation

- Define coefficient vector $\boldsymbol{\beta}^{\mathrm{T}} = (\boldsymbol{\alpha}^{\mathrm{T}}, \boldsymbol{\gamma}^{\mathrm{T}}, \mathbf{b}^{\mathrm{T}})$
- Deviance is $D(\boldsymbol{\beta}) = 2\{l_{\max} - l(\boldsymbol{\beta})\}$
- $\boldsymbol{\beta}$ is estimated by

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} D(\boldsymbol{\beta}) + \sum_j \lambda_j \boldsymbol{\beta}^{\mathrm{T}} \mathbf{S}_j \boldsymbol{\beta} + \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\Sigma}_\theta^- \boldsymbol{\beta} \phi$$

  - $\boldsymbol{\Sigma}_\theta^-$ is a zero padded version of $\boldsymbol{\Sigma}_\theta^{-1}$ so that $\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{\Sigma}_\theta^- \boldsymbol{\beta} = \mathbf{b}^{\mathrm{T}} \boldsymbol{\Sigma}_\theta^{-1} \mathbf{b}$.
  - Similarly the $\mathbf{S}_j$ here are zero padded versions of the originals.
- In practice this optimization is by Penalized IRLS.

Estimating $\boldsymbol{\lambda}$ etc

- We treat $\hat{\boldsymbol{\beta}}$ as a function of $\boldsymbol{\lambda}, \boldsymbol{\theta}, \phi$ for GCV/REML optimization.
- GCV or REML are optimized numerically, with each step requiring evaluation of the corresponding $\hat{\boldsymbol{\beta}}$.
- Note that **computationally** the empirical Bayes approach is equivalent to generalized linear mixed modelling.
- However in most applications it is not appropriate to treat the smooth functions as random quantities to be re-sampled from the prior at each replication of the data set. So the models are not frequentist random effects models.

More inference and degrees of freedom

- The Bayesian approach yields the large sample result

$$\boldsymbol{\beta} \sim N(\hat{\boldsymbol{\beta}}, \mathbf{V}_{\boldsymbol{\beta}})$$

  where $\mathbf{V}_{\boldsymbol{\beta}}$ is a penalized version of the information matrix at $\hat{\boldsymbol{\beta}}$.
- Via an approach pioneered in Nychka (1988, JASA) it is possible to show that CI's based on this result have good frequentist properties.
- Effective degrees of freedom can also be computed. Let $\tilde{\boldsymbol{\beta}}$ denote the *unpenalized* MLE of $\boldsymbol{\beta}$,

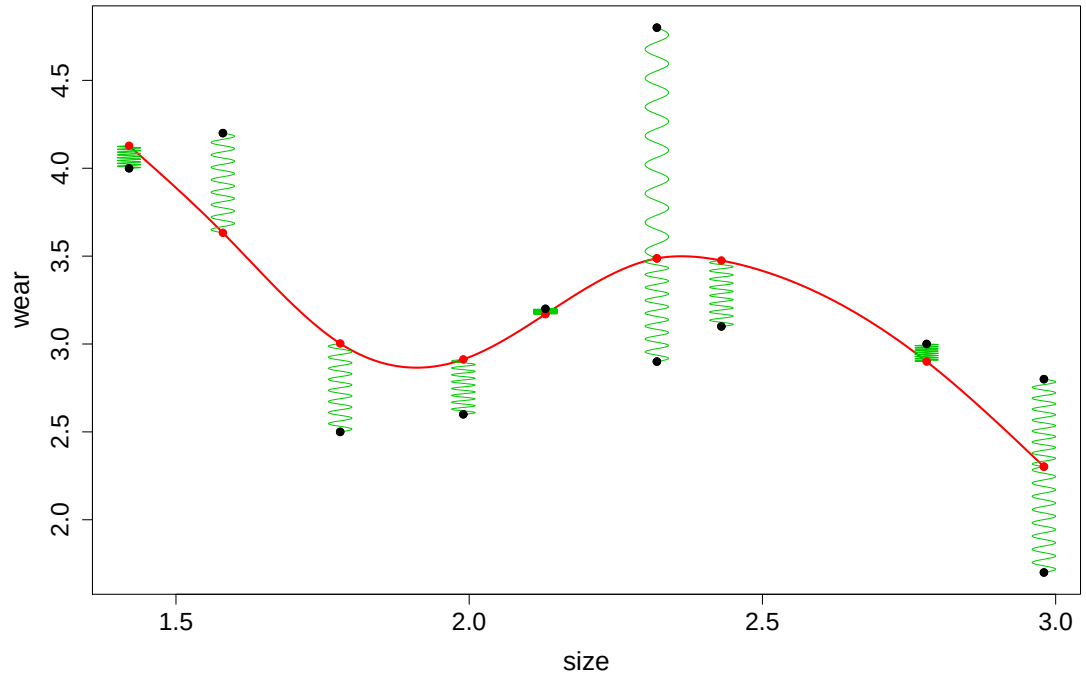
$$EDF = \sum_j \frac{\partial \hat{\boldsymbol{\beta}}_j}{\partial \tilde{\boldsymbol{\beta}}_j}$$

Smooths for semi-parametric GLMs

- To build adequate semi-parametric GLMs requires that we use functions with appropriate properties.
- In one dimension there are several alternatives, and not alot to choose between them.
- In 2 or more dimensions there is a major choice to make.
  - If the arguments of the smooth function are variables which all have the same units (e.g. spatial location variables) then an *isotropic* smooth may be appropriate. This will tend to exhibit the same degree of flexibility in all directions.
  - If the relative scaling of the covariates of the smooth is essentially arbitrary (e.g. they are measured in different units), then *scale invariant* smooths should be used, which do not depend on this relative scaling.

Splines

- Many smooths covered here are based on *splines*. Here's the original underlying idea ...

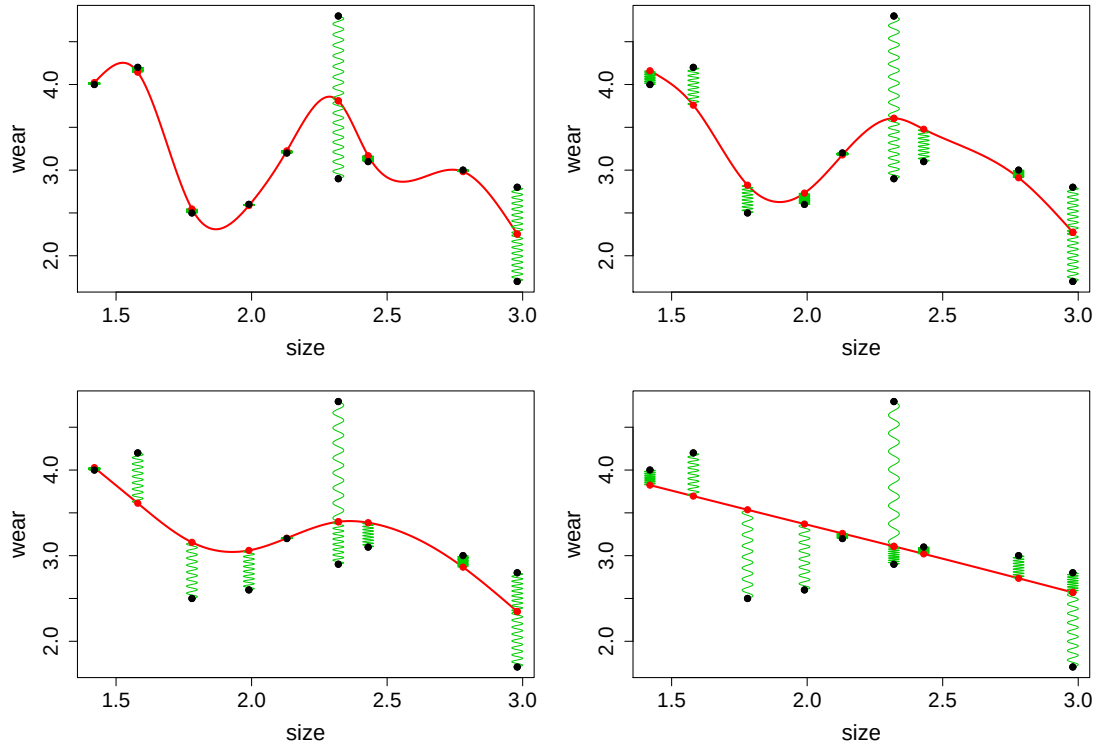


- Mathematically the red curve is the *function* minimizing

$$\sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx.$$

Splines have variable stiffness

- Varying the flexibility of the strip (i.e. varying λ) changes the *spline function* curve.



- But irrespective of λ the spline functions always have the same basis.

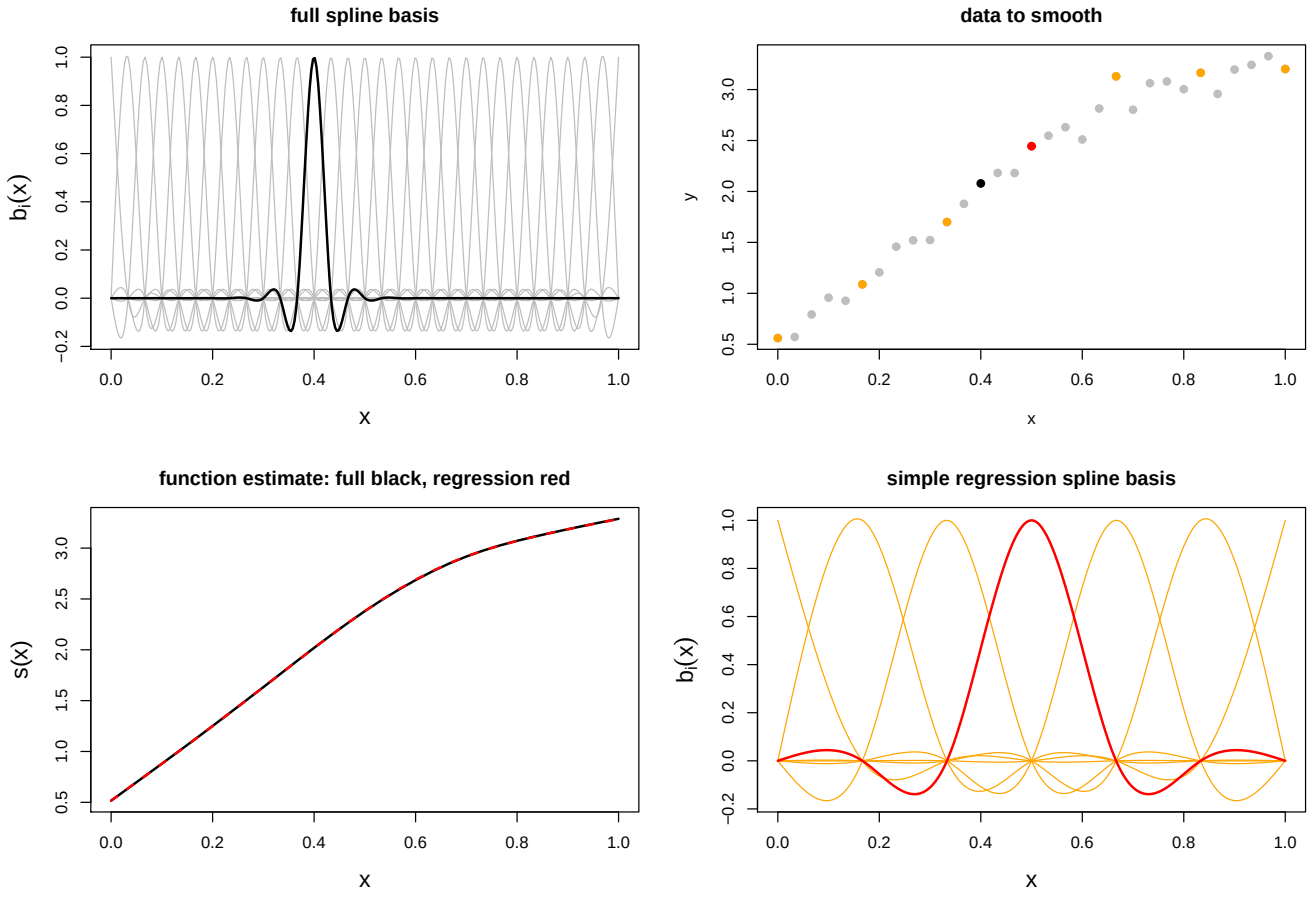
Penalized regression splines

- Full splines have one basis function per data point.
- This is computationally wasteful, when penalization ensures that the *effective* degrees of freedom will be much smaller than this.
- Penalized regression splines simply use fewer spline basis functions. There are two alternatives:
  1. Choose a representative subset of your data (the 'knots'), and create the spline basis as if smoothing only those data. Once you have the basis, use it to smooth all the data.
  2. Choose how many basis functions are to be used and then solve the problem of finding the set of this many basis functions that will optimally approximate a full spline.

  I'll refer to 1 as *knot based* and 2 as *eigen based*.

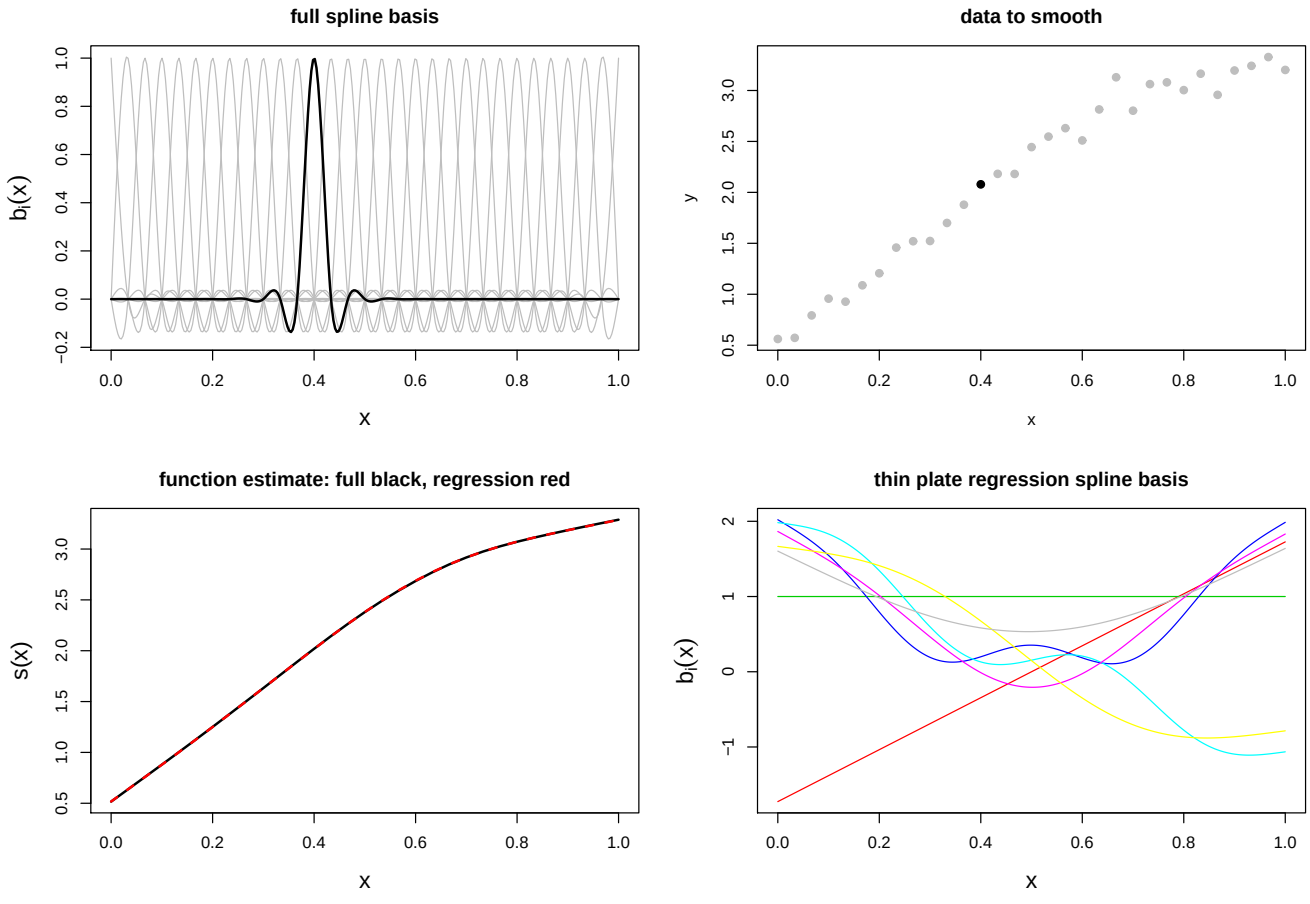
Knot based example: `"cr"`

- In `mgcv` the `"cr"` basis is a knot based approximation to the minimizer of $\sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$ — a cubic spline. `"cc"` is a cyclic version.



Eigen based example: "tp"

- The "tp", *thin plate regression spline* basis is an eigen approximation to a thin plate spline (including cubic spline in 1 dimension).



1 dimensional smoothing in `mgcv`

- Various 1D smoothers are available in `mgcv`...

  `"cr"` knot based cubic regression spline.

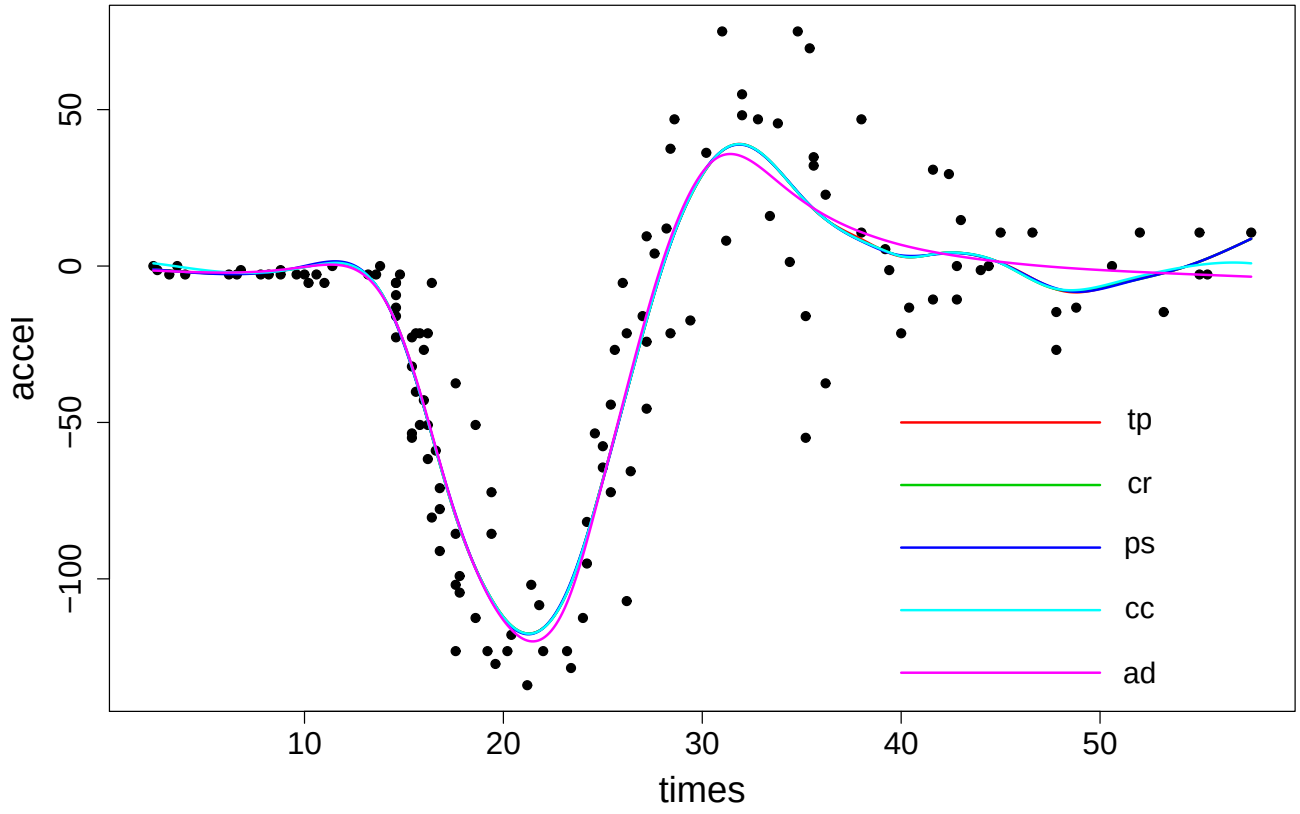
  `"cc"` cyclic version of above.

  `"ps"` Eilers and Marx style p-splines: Evenly spaced B-spline basis, with discrete penalty on coefficients.

  `"ad"` adaptive smoother in which strength of penalty varies with covariate.

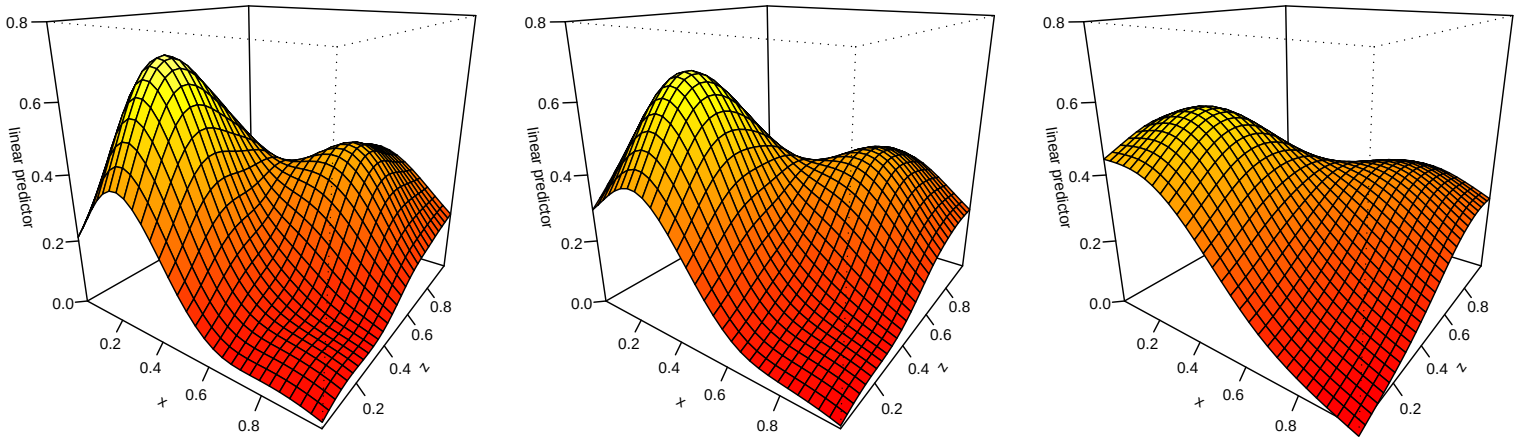
  `"tp"` thin plate regression spline. Optimal low rank eigen approx. to a full spline: flexible order penalty derivative.
- Smooth classes can be added (`?smooth.construct`).

1D smooths compared



Isotropic smooths

- One way of generalizing splines from 1D to several D is to turn the flexible strip into a flexible sheet (hyper sheet).
- This results in a *thin plate spline*. It is an *isotropic* smooth.
- Isotropy may be appropriate when different covariates are naturally on the same scale.
- In `mgcv` terms like `s(x,z)` generate such smooths.



Thin plate spline details

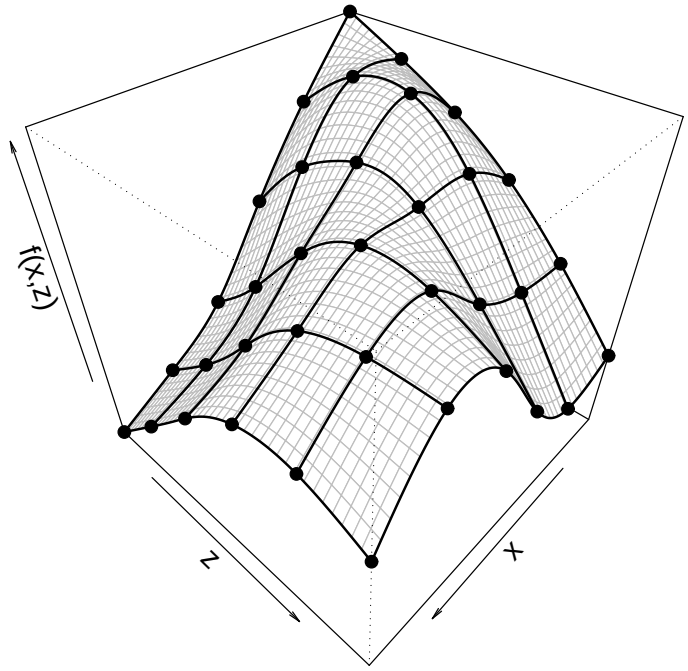
- In 2 dimensions a thin plate spline is the function minimizing

$$\sum_i \{y_i - f(x_i, z_i)\}^2 + \lambda \int f_{xx}^2 + 2f_{xz}^2 + f_{zz}^2 dxdz$$

- This generalizes to any number of dimensions, d, and any order of differential, m, such that $2m > d + 1$.
- Full thin plate spline has n unknown coefficients.
- An optimal low rank eigen approximation is a *thin plate regression spline*.
- A t.p.r.s uses far fewer coefficients than a full spline, and is therefore computationally efficient, while losing little statistical performance.

Scale invariant smoothing: tensor product smooths

- Isotropic smooths assume that a unit change in one variable is equivalent to a unit change in another variable, in terms of function variability.
- When this is not the case, isotropic smooths can be poor.
- *Tensor product smooths* generalize from 1D to several D using a lattice of bendy strips, *with different flexibility in different directions*.



Tensor product smooths

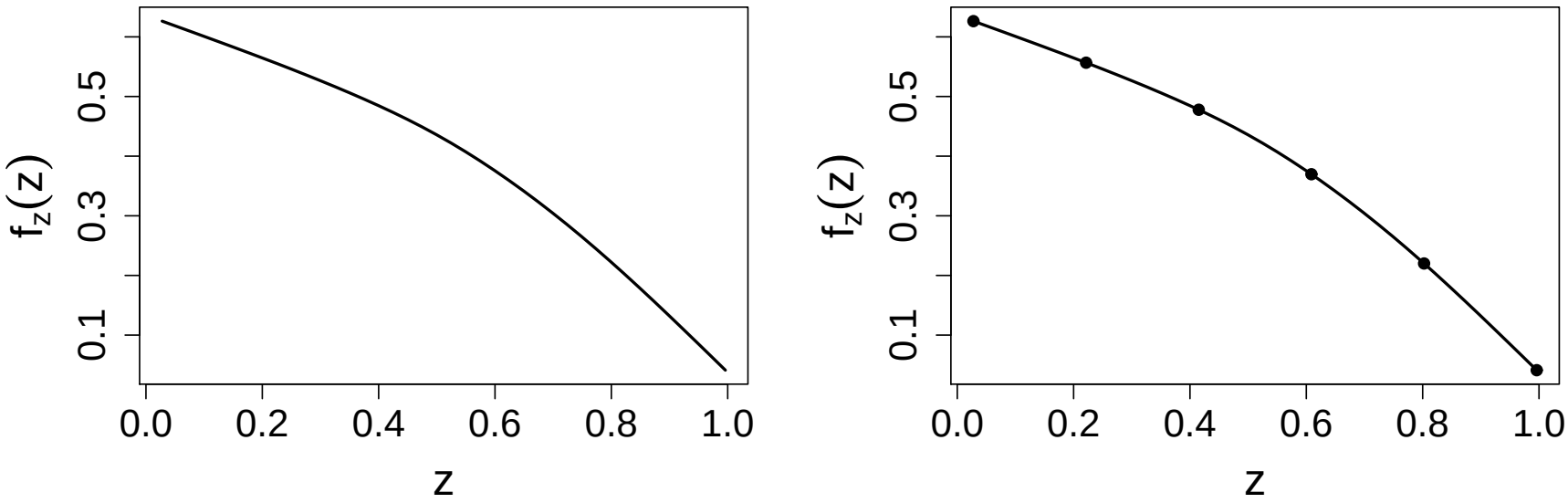
- Carefully constructed tensor product smooths are scale invariant.
- Consider constructing a smooth of x, z.
- Start by choosing *marginal* bases and penalties, as if constructing 1-D smooths of x and z. e.g.

$$f_x(x) = \sum \alpha_i a_i(x), \quad f_z(z) = \sum \beta_j b_j(z),$$

$$J_x(f_x) = \int f_x''(x)^2 dx = \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{S}_x \boldsymbol{\alpha} \;\; \& \;\; J_z(f_z) = \mathcal{B}^{\mathrm{T}} \mathbf{S}_z \mathcal{B}$$

Marginal reparameterization

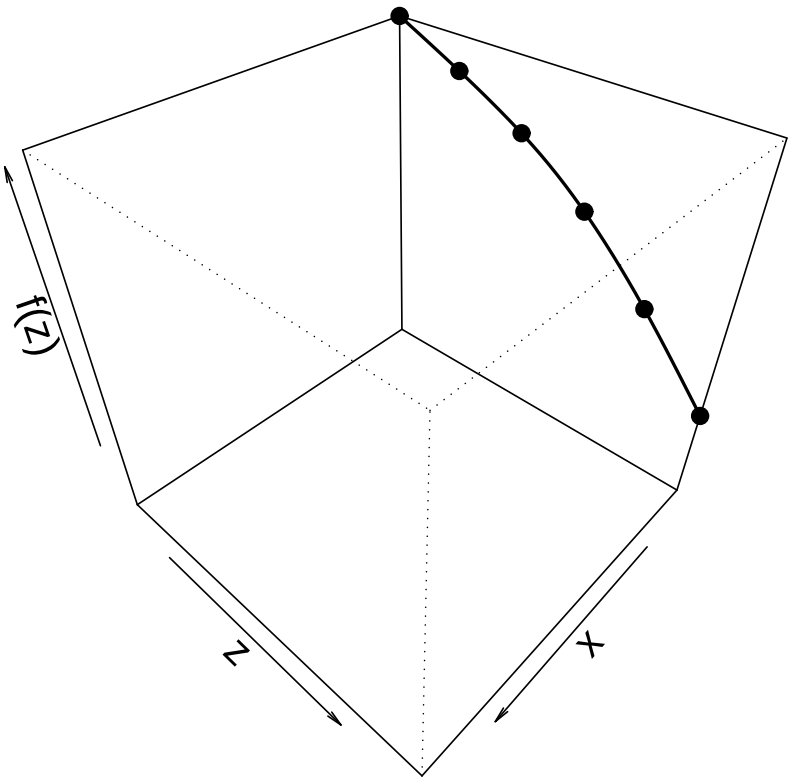
- Suppose we start with $f_z(z) = \sum_{i=1}^{6} \beta_j b_j(z)$, on the left.

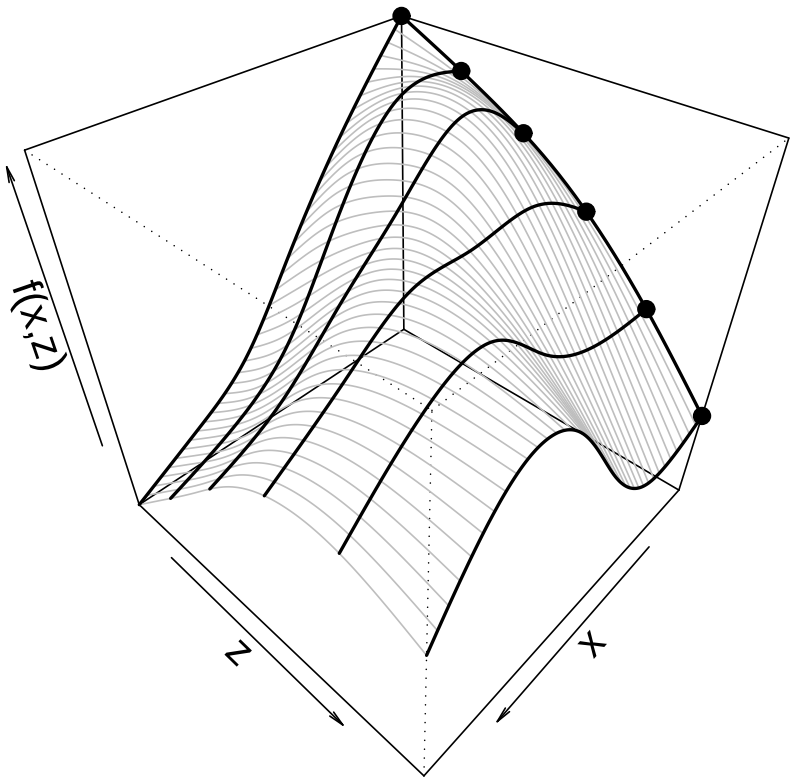


- We can always re-parameterize so that its coefficients are functions heights, at knots (right). Do same for f_x.

Making f_z depend on x

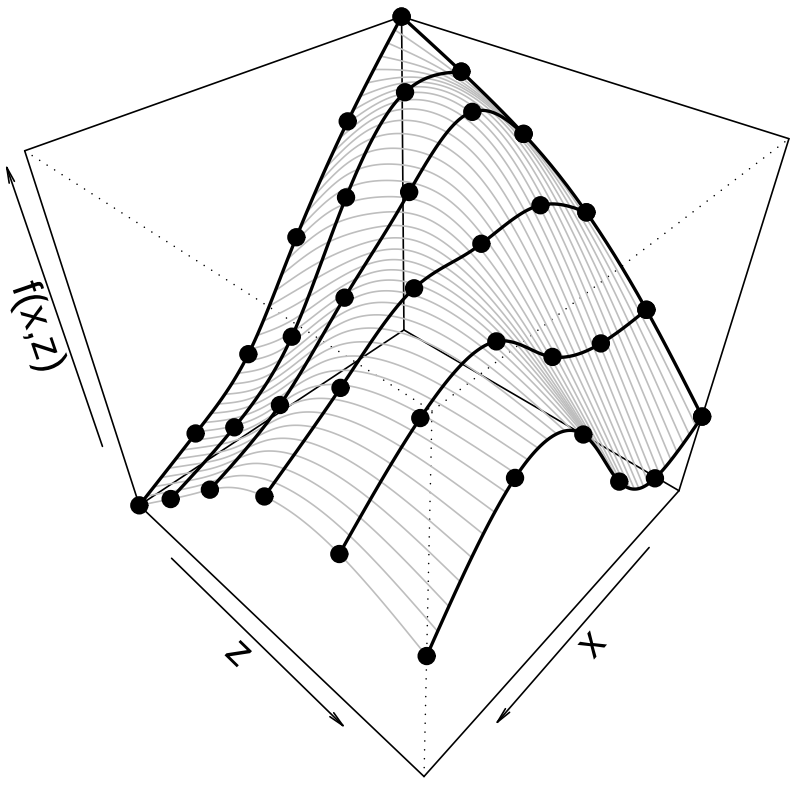
- Can make f_z a function of x by letting its coefficients vary smoothly with x

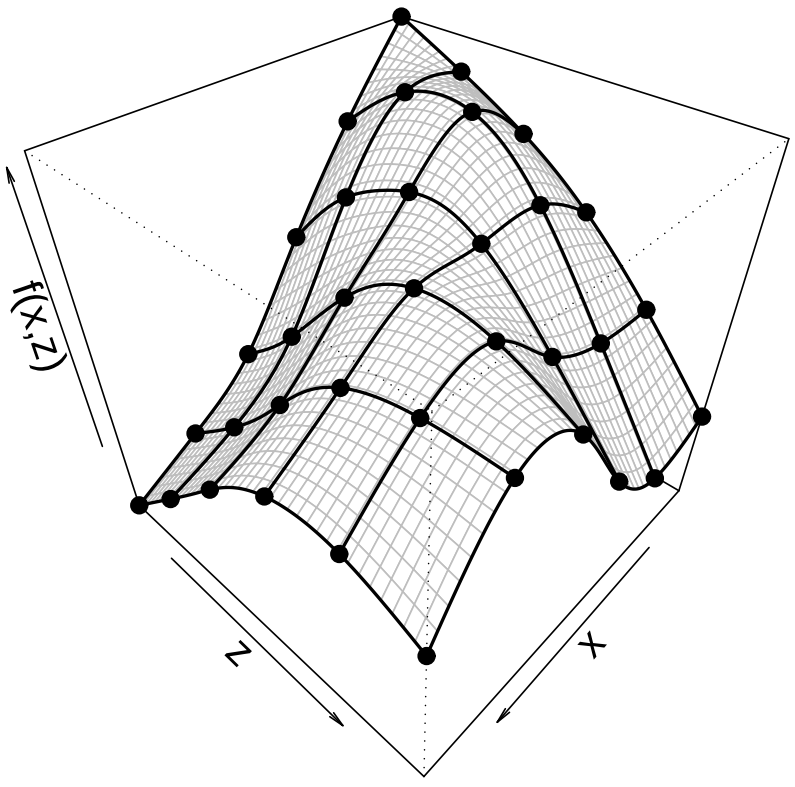




The complete tensor product smooth

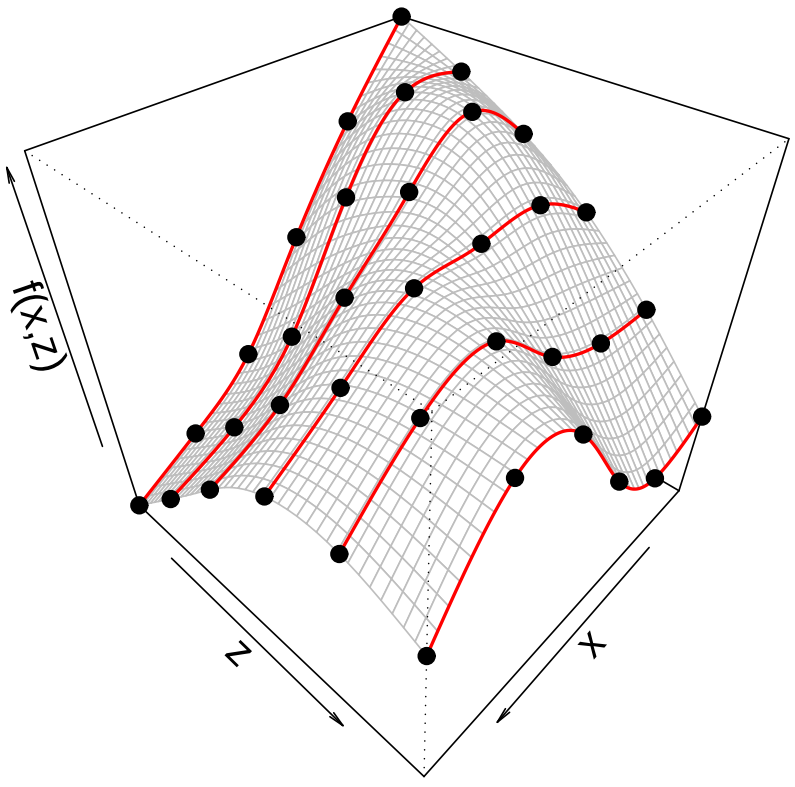
- Use f_x basis to let f_z coefficients vary smoothly (left).
- Construct in symmetric (see right).

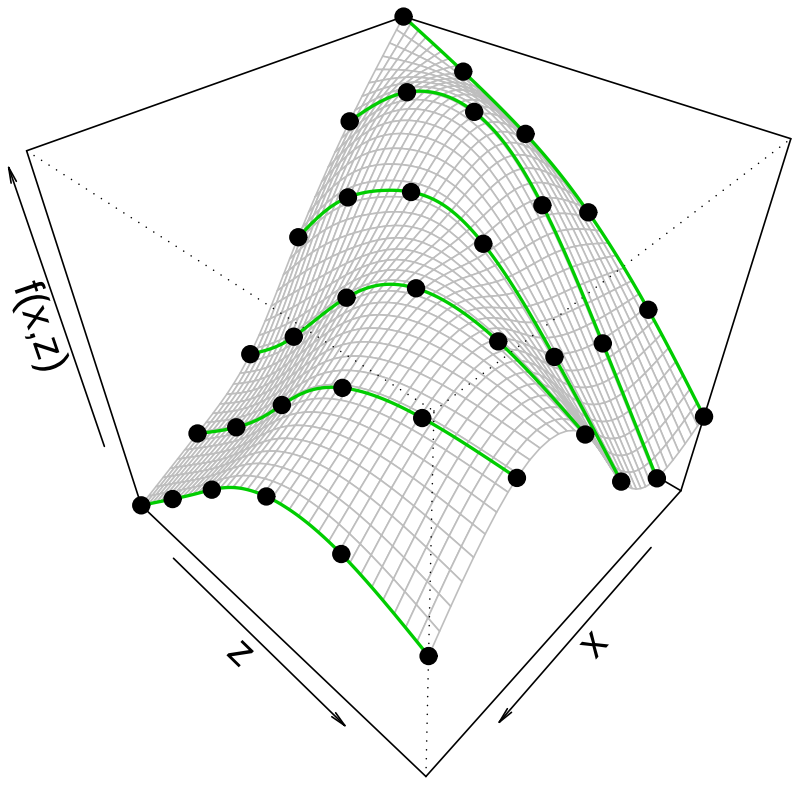




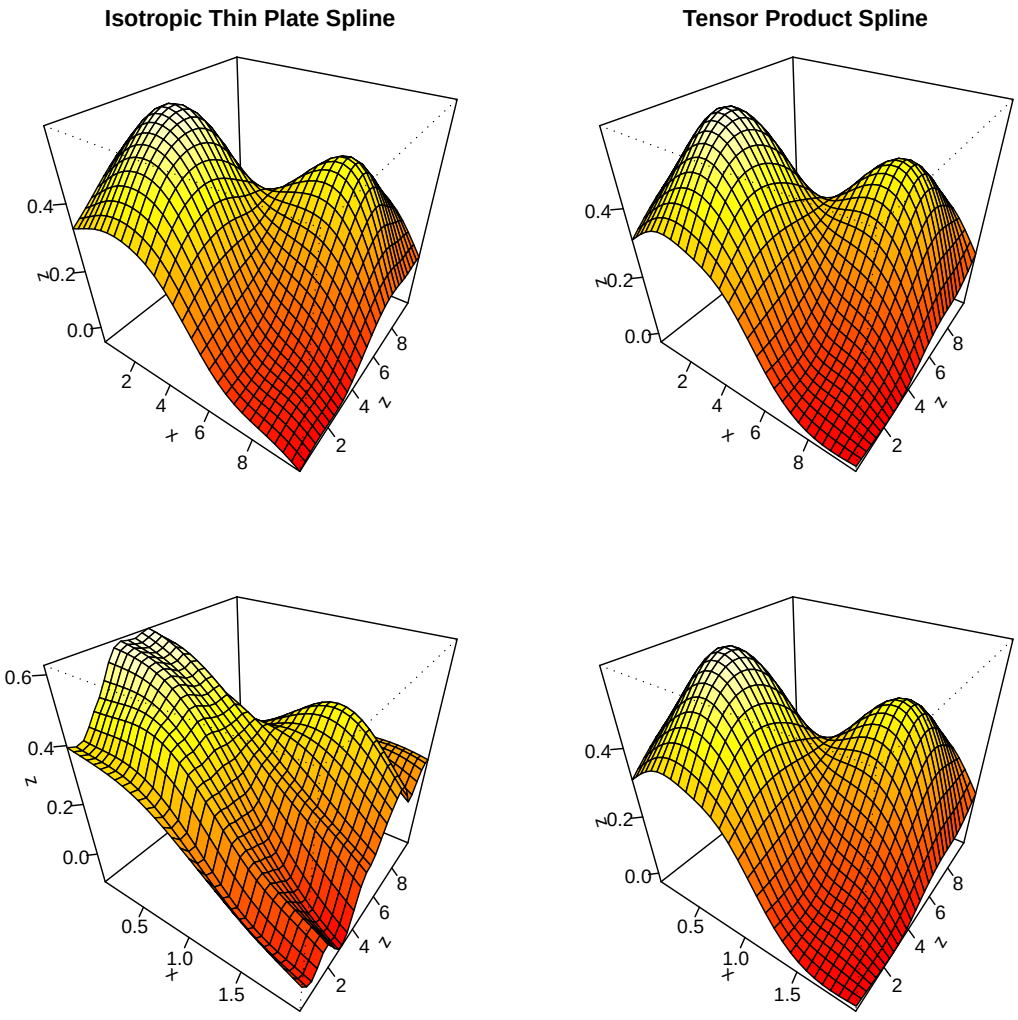
Tensor product penalties - one per dimension

- x-wiggliness: sum marginal x penalties over red curves.
- z-wiggliness: sum marginal z penalties over green curves.





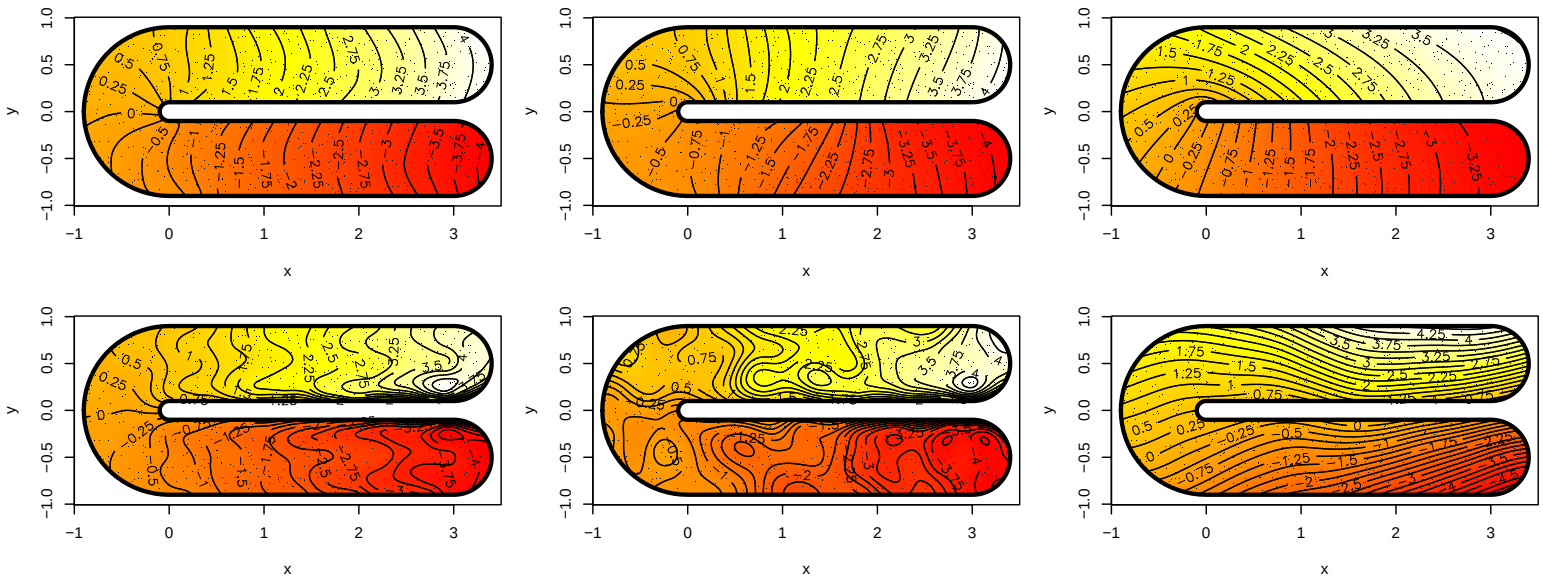
Isotropic vs. tensor product comparison



. . . each figure smooths the same data. The only modification is that x has been divided by 5 in the bottom row.

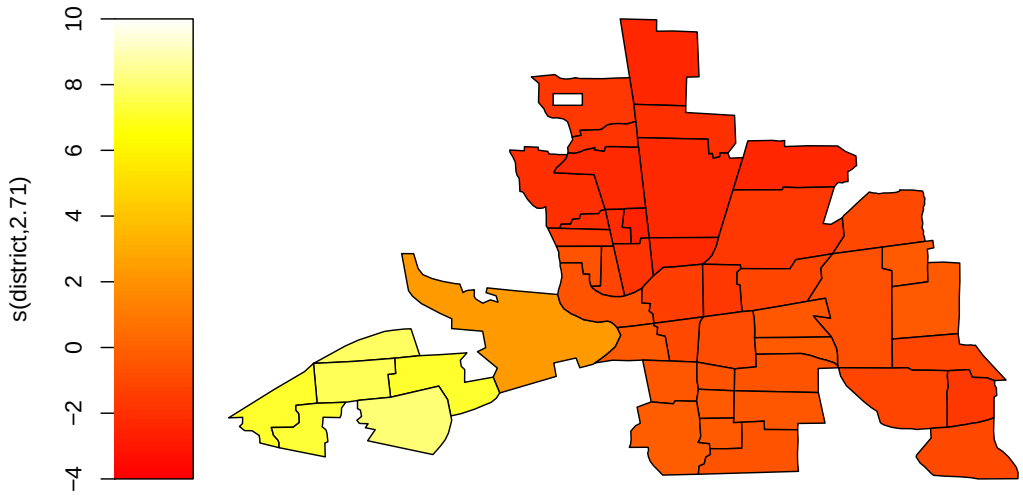
Soap film smoothing

- Sometimes we require a smoother defined over a bounded region, where it is important not to smooth across boundary features.
- A useful smoother can be constructed by considering a distorted soap film suspended from a flexible boundary wire.



Markov Random fields

- Sometimes data are associated with discrete geographic regions.
- A neighbourhood structure on these regions can be used to define a Markov random field, which induces a simple penalty structure on region specific coefficients.



0 Dimensional smooths

- Having considered smoothing with respect to 1, 2 and several covariates, or a neighbourhood structure, consider the case where we want to smooth without reference to a covariate (or graph).
- This amounts to simply shrinking some model coefficients towards zero, as in ridge regression.
- In the context of penalized GLMs such zero dimensional smooths are equivalent to simple Gaussian random effects.

Specifying GAMs in R with mgcv

- `library(mgcv)` loads a semi-parametric GLM package.
- `gam(formula,family)` is quite like `glm`.
- The `family` argument specifies the distribution and link function. e.g. `Gamma(log)`.
- The response variable and linear predictor structure are specified in the model `formula`.
- Response and parametric terms exactly as for `lm` or `glm`.
- Smooth functions are specified by `s` or `te` terms. e.g.

$$\log\{\mathbb{E}(y_i)\} = \alpha + \beta x_i + f(z_i), \; y_i \sim \text{Gamma},$$

is specified by...

```
gam(y ~ x + s(z),family=Gamma(link=log))
```

More specification: `by` variables

- Smooth terms can accept a `by` variable argument, which allows $L_{ij} f_j$ terms other than just $f_j(x_i)$.
- e.g. $\mathbb{E}(y_i) = f_1(z_i)w_i + f_2(v_i)$, $y_i \sim$ Gaussian, becomes

  ```
  gam(y ~ s(z,by=w) + f(v))
  ```

  i.e. $f_1(z_i)$ is multiplied *by* w_i in the linear predictor.
- e.g. $\mathbb{E}(y_i) = f_j(x_i, z_i)$ if factor g_i is of level j, becomes

  ```
  gam(y ~ s(x,z,by=g) + g)
  ```

  i.e. there is one smooth function for each level of factor variable `g`, with each y_i depending on just one of these functions.

Yet more specification: a summation convention

- `s` and `te` smooth terms accept *matrix* arguments and `by` variables to implement general $L_{ij}f_j$ terms.
- If **X** and **L** are $n \times p$ matrices then

  ```
  s(X,by=L)
  ```

  evaluates $L_{ij}f_j = \sum_k f(X_{ik})L_{ik}$ for all i.
- For example, consider data $y_i \sim \text{Poi}$ where

$$\log\{\mathbb{E}(y_i)\} = \int k_i(x)f(x)dx \simeq \frac{1}{h}\sum_{k=1}^{p} k_i(x_k)f(x_k)$$

  (the x_k are evenly spaced points).
- Let $X_{ik} = x_k \; \forall \; i$ and $L_{ik} = k_i(x_k)/h$. The model is fit by

  ```
  gam(y ~ s(X,by=L),poisson)
  ```